PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30, 17/27, G11C 13/04 — A1

(11) International Publication Number: WO 97/12334

(43) International Publication Date: 3 April 1997 (03.04.97)

(21) International Application Number: PCT/US96/15624

(22) International Filing Date: 25 September 1996 (25.09.96)

(30) Priority Data:
08/533,663 25 September 1995 (25.09.95) US

(71) Applicant: INTERNATIONAL COMPU RESEARCH, INC. [US/US]; 1401 Dove Street #500, Newport Beach, CA 92660 (US).

(72) Inventors: SMITH, Joseph, C.; 4666 Northwest Marine Drive, Vancouver, British Columbia V6R 1B9 (CA). McCLEAN, John, A.; 1834 Beacon Street #15, Brookline, MA 92146 (US). ATHERTON, Bruce, C.; 5229 Elgin Street, Vancouver, British Columbia V5W 3J9 (CA).

(74) Agent: LAND, John; Fish & Richardson P.C., Suite 1400, 4225 Executive Square, La Jolla, CA 92037 (US).

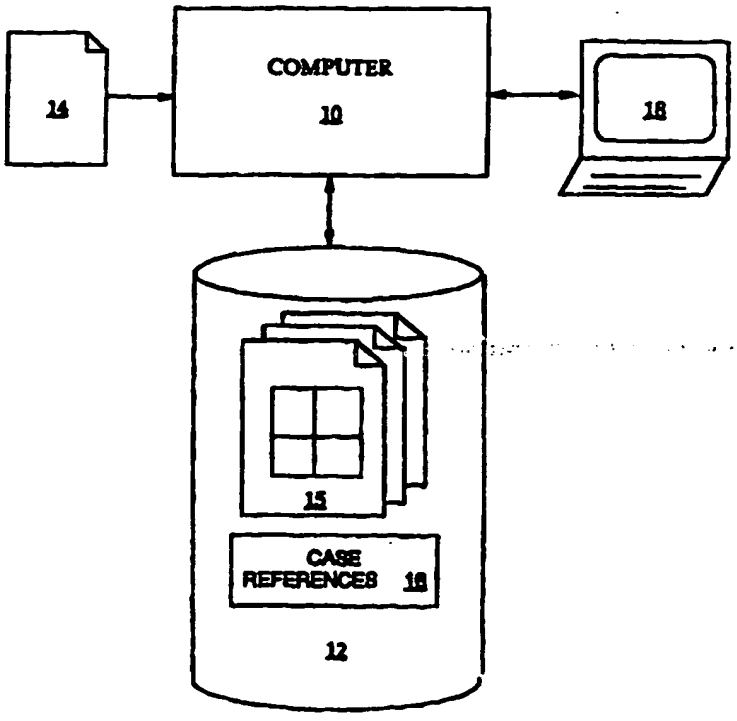
(81) Designated States: CA, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: MATCHING AND RANKING LEGAL CITATIONS

(57) Abstract

For use in a system for retrieval of legal information by data processing systems, methods for matching a case identified in the parsing (210, 212) of a legal text within a database (15-16) of case references (16), and methods for ranking (316) the relevance of cases matching (216) a search query.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	**GB**	United Kingdom	**MW**	Malawi
AT	Austria	**GE**	Georgia	**MX**	Mexico
AU	Australia	**GN**	Guinea	**NE**	Niger
BB	Barbados	**GR**	Greece	**NL**	Netherlands
BE	Belgium	**HU**	Hungary	**NO**	Norway
BF	Burkina Faso	**IE**	Ireland	**NZ**	New Zealand
BG	Bulgaria	**IT**	Italy	**PL**	Poland
BJ	Benin	**JP**	Japan	**PT**	Portugal
BR	Brazil	**KE**	Kenya	**RO**	Romania
BY	Belarus	**KG**	Kyrgystan	**RU**	Russian Federation
CA	Canada	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CF	Central African Republic			**SE**	Sweden
CG	Congo	**KR**	Republic of Korea	**SG**	Singapore
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovakia
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LR**	Liberia	**SZ**	Swaziland
CS	Czechoslovakia	**LT**	Lithuania	**TD**	Chad
CZ	Czech Republic	**LU**	Luxembourg	**TG**	Togo
DE	Germany	**LV**	Latvia	**TJ**	Tajikistan
DK	Denmark	**MC**	Monaco	**TT**	Trinidad and Tobago
EE	Estonia	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**UG**	Uganda
FI	Finland	**ML**	Mali	**US**	United States of America
FR	France	**MN**	Mongolia	**UZ**	Uzbekistan
GA	Gabon	**MR**	Mauritania	**VN**	Viet Nam

MATCHING AND RANKING LEGAL CITATIONS

BACKGROUND OF THE INVENTION

1. *Field of the Invention*

The invention relates to retrieval of legal information by data processing systems.

2. *Background*

The essence of legal research in common law jurisdictions is the retrieval of relevant legal information. Within the world of the common law, cases are one of the most important sources of legal information. Unlike legislation, they have no official indexing or form of informational organization. Legal cases, as produced by the courts, generally lack defined structural elements such as titles, abstracts and subsections. Legal publishers and some legal electronic information vendors devote a great deal of resources towards the indexing, summarization and classification of cases and other documents. Private sector publishers often place this added information at the front of a case in the form of what has become known as a headnote. This permits the user to quickly ascertain whether or not the document is likely to be relevant. These headnotes are prepared by legally trained people, at a significant cost, and the publishers retain copyright in them. Without headnotes, a researcher may have to spend a great deal of time reading through a case in order to ascertain whether or not it is of interest.

An ideal legal information system ought to receive queries in a very simple form that allows the user to specify the issues of interest and to intelligently identify documents

addressing the issues specified by the query. The user ought to be able to look at the first screen of a retrieved document and come to some conclusion as to the relevancy of it in regard to the search query. Returned documents ought to be ordered in terms of the degree of relevancy to the query. The system should be rich in relevant hypertext links.

The law is generally found in four different kinds of documents. The law is found in a non-authoritative form in legal treatises. It is found in authoritative form in the precedents and written decisions of the courts. It is found in legislative fiats such as statutes, codes and regulations. And it is found in a non-authoritative form in the framework of facts and arguments which define actual litigation.

Coinciding with these kinds of sources, one can create a profile of any legal document in terms of concepts, case citations, statute citations, and facts. Such a profile of a legal document can be used throughout an information system to form queries, to summarize the document, and to place hyper-text links automatically between the summaries and the full text of the document.

A case, as a legal document, can be represented in terms of a profile of (i) the relevant important concepts, (ii) the citation to the leading cases on the topic, (iii) the relevant legislative provisions, if applicable, and (iv) the significant factual terms. Thus a case dealing with a felony murder where a convenience store owner dies of a heart attack shortly after a robbery could be represented by the profile shown in the following table.

CONCEPTS felony-murder	CITED CASES People v. Cahill (1995) 22 Cal.App.4th 296
CITED STATUTES Crim. Code § 187	FACTS convenience store robbery heart attack

Different cases dealing with the same legal issues but different fact patterns can be represented by changing the significant terms of the facts quadrant. In the same way, the terms of the facts quadrant can remain fixed, but different concepts, case citations, and statutory provisions can be used to represent cases on similar facts but dealt with in terms of different legal doctrines.

A legal text management tool that exploits this form of representation of legal material is FLEXICON (Fast Legal EXpert Information CONsultant), which has been described, for example, in the following articles, the disclosures of which are incorporated herein by this reference: Gelbart, D. et al., "Flexicon: An Evaluation Of A Statistical Ranking Model Adapted To Intelligent Legal Text Management", Proceedings of the Fourth International Conference on Artificial Intelligence and Law, University of British Columbia, Vancouver, Canada (1993); Gelbart, D. et al., "Beyond Boolean Search: FLEXICON, A Legal Text-Based Intelligent System", Proceedings of the Third Conference on Artificial Intelligence & Law, University of British Columbia, Vancouver, Canada, pp. 225-234 (1991); Gelbart, D. et al., "Toward A Comprehensive Legal Information Retrieval System", Database and Expert Systems Applications, Proceedings of the International Conference, Vienna, Austria, pp. 121-125 (1990); Gelbart, D. et al., "Current Issues in Text Retrieval:

FLEXICON, A Legal Text-Based Intelligent System", University of British Columbia, Faculty of Law Artificial Intelligence Research Project, Vancouver, Canada, pp. 1-5 (1989); Gelbart, D. et al., "Towards Combining Automated Text Retrieval and Case-Based Expert Legal Advice", Law Technology Journal, CTI Law Technology Centre and Bileta, Vol. 1, No. 2, pp. 19-24 (1992); Gelbart, D. et al., "Effective Legal Information Retrieval Systems", Council of Canada and The Law Foundation of British Columbia; and Gelbart, D. et al., "Flexicon, A New Legal Information Retrieval System", Canadian Law Libraries, Bibliotheques de Droit Canadiennes, Vol. 16, No. 1, pp. 9-12 (1991).

SUMMARY OF THE INVENTION

In a system such as FLEXICON, significant preprocessing of the cases or other legal documents is necessary to extract the information required to fill out the documents' profiles. The invention provides computer-implemented methods useful in automating this preprocessing. Complementary to this preprocessing, the invention also provides methods useful in ranking the results of a search where the query is formulated in terms of document profile categories.

In general, in one aspect, the invention provides a method for matching a current case reference to a set of case references. The method includes maintaining a database of case references that have been processed by the method, parsing the current reference for its citations, and for each citation, parsing a citation into its volume, reporter, and page, and searching the database for a matching case by applying a set of tests to the cases in the database, the tests including: if a candidate case matches the current case in two volume-reporter-page citations from different

reporters, the candidate case is a match. In another aspect, the invention includes parsing the current case reference for its party names, and for each non-noise word in each party name, acquiring a sound-alike value for the non-noise word; and searching the database for a matching case by applying a set of tests including: if a candidate case matches the current case in one citation, and the cases' names match at least loosely, and neither case has a citation that is inconsistent with the citations of the other case, the candidate case is a match.

In another aspect, the invention includes the steps of initializing a candidates set to be empty; adding to the candidate set a case reference from the database if it has a same party name as the current reference, by sound-alike values; adding to the candidate set a case reference from the database if the case reference has a volume-reporter-page citation matching the current citation; and searching the candidate set rather than the entire database of case references for a matching case by applying a set of tests to the cases in the candidate set. In another aspect, the method includes applying a set of tests including: if a candidate case matches the current case in two volume-reporter-page citations from different reporters and the court information for the two cases is not inconsistent, the candidate case is a match; if a candidate case matches the current case in two volume-reporter-page citations from different reporters, and the year information is not inconsistent, the candidate case is a match; and if a candidate case matches the current case in one citation and both the court information and the year information for the two cases is not inconsistent, and neither case has a citation that is inconsistent with the citations of the other case, the candidate case is a match. In another aspect, the invention includes applying tests including: if a candidate case matches the current case in one citation, and the courts and the years also match, the cases' names very tightly, and the cases have less than two citations that are inconsistent from one case to the other, the candidate is

match; and, if a candidate case matches the current case in both courts and years, the cases' names very tightly, and neither case has a citation that is inconsistent with the citations of the other case, the candidate case is match.

In general, in another aspect, the invention provides a method for ranking the relevance of a target document found by a search query in a set of documents. The method includes providing a set of weighting factors defined by user for the search query, at least one of which weighting factors differs from the others and at least one of which weighting factor has a negative value; applying a metric function to the search query, the weighting factors, and the target document to produce a similarity measure; and ranking the target document by its similarity measure. In another aspect, the method includes the calculation of an inner product as part of the metric function.

In general, in another aspect, the invention provides a method for ranking the relevance of a target document found by a search query in a set of documents, where the document terms and the search terms are each of one of the plurality of types. The method includes, for each type in the plurality of types, applying a metric function to those terms of the search query and the target document having that type, to produce a type-based similarity measure; combining the type-based similarity measures by applying a user-selectable type-based weight to each of the target document's type-based similarity measures to produce a final similarity measure; and ranking the target document by the final similarity measure. In another aspect, the method includes combining the type-based similarity measures and applying a factor to the similarity measures based on number of different types for which some search term matched the target document. In another aspect, the method includes providing a set of weighting factor for each search query, at lease one of which weighting

factors differs from the others, and applying a metric function to the search query, the weighting factors, and the target document to produce a similarity measure.

The invention has a number of advantages.

For example, the case matching method provides ideal case citations (case references) that improve the accuracy and usefulness of case name- and citation-based searches and hypertext links and that, when used with automatic case extraction, in affect correct extraction errors. The case matching method, by recognizing references to the same case, improves the usefulness of statistics that characterize a legal text by the frequency with which it cites cases. Similarly, the method improves the quality of statistics taken on a database of cases as a whole. Also, the search result ranking method improves the usefulness of search results by allow the user (requestor) to weigh the importance of term in a search request and to increase the ranking of documents that match on multiple quadrants.

Other advantages and features will become apparent from the following description and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in, and constitute a part of, the specification, schematically illustrate specific embodiments of the invention and, together with the general description given above and the detailed description of the embodiments given below, serve to explain the principles of the invention.

FIGURE 1 is a block diagram of a data processing arrangement supporting a method for processing legal text.

FIGURE 2 is a flowchart of a method for matching case references.

FIGURES 3 is a flowchart of a method for ranking cases found in a search.

DETAILED DESCRIPTION

Turning to FIGURE 1, a preprocessor for a legal text management system is implemented as a computer program on a computer 10 coupled to mass data store 12 on which are stored one or more sets of data, preferably stored in one or more databases managed by a database management system (not shown). Also, available for use by a user (generally human, but possibly also an application program) is a computer interface 18 through which search requests can be made of the data on data store 12, and through which information can be provided to the user.

The preprocessor takes as input a document 14 (which may be a data file, part of a data file, or other form of text input) containing the text of a judicial opinion (a case) or other legal text. (For clarity of exposition, the following description is given solely in terms of a case; however, it will be clear that the methods described are applicable to the processing of other kinds of legal material.)

Document 14 may have formatting information (such as underlining marks) in it in addition to plain text. Such information may be useful in preprocessing the document -- for example, the names of cases cited in judicial opinions are often

written in italics or underlined – but the methods described herein are intended for fully generally application and so do not rely on such non-textual information.

The preprocessor recognizes and extracts from the text of document 14 case references and statute references, as will be described. A full case reference normally has a case name, which is normally formed from the names of the parties (e.g., "Lochner v. New York"), one or more citations, each indicating a volume number, reporter, and page number (e.g., "198 U.S. 45"), and a date. (The term "citation" is used in two senses: first, to refer to the entire reference, including the case name; and second, more specifically, to refer to the volume-reporter-page information. Where appropriate to avoid ambiguity, the unidiomatic terms "case reference" or "reference" will be used to denote the former.)

Turning to FIGURE 2, the preprocessor parses the case name at step 210 and the case citation (or citations) at step 212. Cited cases (case references) are recognized by the preprocessor by means of template matching. Both the form of the case names ("v."and "In re" primarily) and the form of the case citations (the familiar "volume, reporter, page" sequence) are matched by different template matching mechanisms. If a portion of text matches a case name template, the preprocessor searches subsequent to the name for citations to reporters. If a reporter citation is found first, the preceding text is examined to see if it contains a case name in a form not recognized by the general purpose case name template matcher. This process exploits the conventional use of capitalization. The case names are parsed into names of parties; and the citations are parsed into volume, reporter, page, court, and year, to the extent the information appears in the case reference.

Once a case has been found in a document, subsequent references to the case in the document by means of shortened forms of the case name (e.g., ". . . in the Smith case, supra, . . .")are recognized. (The preprocessor assumes that a party name from a previously cited case (unless it is the name of a party in the present case), when it appears in the text, is a reference to the previously cited case.)

When a case reference has been extracted, it is added to a database 16 of case references, as will be described. As each new reference is added to the database, a check is made for a match with the references already in the database. If one is found, steps 216 and 218, any new information about the case from the current reference (e.g.,a parallel citation to an unofficial reporter) is added to the database record at step 220; otherwise, a new record is made for the reference at step 222. (The record for a reference may be stored as a single record, in the technical database sense, but for purposes of this description the term should be understood more broadly to include any coherent set of data, stored and accessible in the database, that includes information about the case to which the case reference refers.) At optional step 214, a subset of the database -- a candidate set of cases -- is extracted before matching is done at step 216, to include cases that match on any of the parts of the case reference. The extraction of a candidate set is done, in one embodiment, only for the sound-alike terms (which will be described), for the sake of efficiency.

The matching of case references (step 216) against the database 16 is done with heuristic algorithms that rely principally on citations and secondarily on names. Thus, for each citation in the reference, a search may be made of the database and the results evaluated by the case matching filter, described below. If no match is

found for one citation, the next citation, if any, of the reference is considered until all citations have been tried.

If no matches are found based on the reference's citation(s), the preprocessor filter (step 216) tries the reference's party names, as follows. A sound-alike value is created for each non-noise word of each party name using, for example, any one of the class of algorithms commonly known as "soundex algorithms". These associate a unique number (which will be called the "soundex value") with words that sound alike. (A frequently used example is "Smith" and "Smythe".) The database record for a case reference includes a soundex value for each non-noise part of each party name. The party name soundex values for the current reference are used to search the database. The records retrieved are sorted based on number of matches, and some number of the candidates from the top of the sort order are tested with the case matching filter until either a match is found or the candidates are exhausted.

The main module of a case matching filter is implemented in the C++ method same_case set forth in the table below, and in the routines it calls, whose functions are described by their names and the comments associated with them.

```
/* This function returns TRUE if the two cases passed to it are the same
and otherwise it returns FALSE. */

int     case_table_type::same_case(full_case_info_type *x,
case_info_type *y)
{
int       common_cites;
int       same_date;
int       same_court;
int       name_match;
int       inconsistent_cites;
int       all_cites_match;
```

```
common_cites = cites_in_common(x, y, &inconsistent_cites,
&all_cites_match);
same_date = compare_dates(x, y);
same_court = compare_courts(x, y);
name_match = compare_names(x, y);

if (common_cites >= 2 && same_court)
        return(TRUE);
if (common_cites >= 2 && same_date)
        return(TRUE);
if (common_cites == 1 && name_match >= LOOSE && inconsistent_cites < 1)
        return(TRUE);
if (common_cites == 1 && same_date && same_court && name_match ==
VERY_TIGHT && inconsistent_cites < 2)
        return(TRUE);
if (common_cites == 1 && same_date && same_court && all_cites_match)
        return(TRUE);
if (name_match == VERY_TIGHT && same_date && same_court &&
inconsistent_cites < 1)
        return(TRUE);
return(FALSE);
}

/* This function compares the dates of two cases and returns TRUE if
they are the same and FALSE otherwise. */

int     case_table_type::compare_dates(full_case_info_type *x,
case_info_type *y)
{
if (x->year == y->year || y->year == UNKNOWN || x->year == UNKNOWN)
        return(TRUE);
return(FALSE);
}

/* This function returns the number of cites the two cases passed to it
have in common, and also returns the number of inconsistent cites the
two cases have in common. Inconsistent cites are two different cites to
the same reporter. If the case with the lesser number of cites all match
cites of the other case, then all_match is TRUE, otherwise FALSE. */

int     case_table_type::cites_in_common(full_case_info_type *x,
case_info_type *y, int *incon_cites, int *all_match)
{
int                     common_cites = 0;
int                     inconsistent_cites = FALSE;
cite_type               x_cites[MAX_CITES], y_cites[MAX_CITES];
int                     xi = 0;
int                     yi = 0;
int                     min_no_cites;
streamoff               y_pos;
disk_link_cite_type     y_ct;
ram_link_cite_type      *rl;

rl = &(x->cite);
while(rl != NULL && rl->cite.valid()) { // read x cites
        x_cites[xi] = rl->cite;
        rl = rl->next;
```

```
            xi++;
            }
        x_cites[xi].invalidate();

        y_pos = y->cites;                    // read y cites
        while (y_pos != END) {
            cite_ptr_file.seekg(y_pos, ios::beg);
            cite_ptr_file >> y_ct;
            y_cites[yi] = y_ct.cite;
            y_pos = y_ct.next_cite;
            yi++;
            }
        y_cites[yi].invalidate();

        min_no_cites = (xi < yi) ? xi : yi;
        if (xi != 0 || yi != 0)
            inconsistent_cites = CITES_EXIST;
        xi = yi = 0;                                    // compare cites
        while(x_cites[xi].valid()) {
            while(y_cites[yi].valid()) {
                if (x_cites[xi] == y_cites[yi]) {
                    common_cites++;
                    }
                else if ((x_cites[xi].normal.unused == NO_YEAR &&
        y_cites[yi].normal.unused == NO_YEAR && x_cites[xi].normal.reporter ==
        y_cites[yi].normal.reporter) ||
                         (x_cites[xi].normal.unused != NO_YEAR &&
        y_cites[yi].normal.unused != NO_YEAR && x_cites[xi].normal.reporter ==
        y_cites[yi].normal.reporter)) {
                    inconsistent_cites = TRUE;
                    }
                yi++;
                }
            yi = 0;
            xi++;
            }
        if (x_cites[0].valid() && y_cites[0].valid() && common_cites == 0)
            inconsistent_cites = NO_CITE_MATCH;
        *incon_cites = inconsistent_cites;
        if (min_no_cites > 0 && min_no_cites == common_cites)
            *all_match = TRUE;
        else
            *all_match = FALSE;
        return(common_cites);
        }

        /* This function returns TRUE if two courts are the same and FALSE
        otherwise. Two courts are counted as being the same if all of the
        shorter court abbreviations letters are found in the same order in the
        longer court abbreviation. Therefore 'B.C.S.C.' would match with 'S.C.'
        or 'SC'. */

        int     case_table_type::compare_courts(full_case_info_type *x,
        case_info_type *y)
        {
        char     ct[MAX_COURT_LEN];
        char   *cp1, *cp2;
```

```
if (y->court == NO_COURT || x->court[0] == '\0')
        return(TRUE);
if (y->court == court->court_number(x->court))
        return(TRUE);
court->court_string(ct, y->court);
cp1 = ct;
cp2 = x->court;
if (number_letters(cp1) > number_letters(cp2)) {
        cp1 = x->court;
        cp2 = ct;
        }
while(*cp1 != '\0') {
        if (*cp2 == '\0')
              return(FALSE);
        if (*cp1 == *cp2) {
              cp1++;
              cp2++;
              }
        else {
              cp2++;
              }
        while (*cp1 == ' ' || *cp1 == '.')
              cp1++;
        while (*cp2 == ' ' || *cp2 == '.')
              cp2++;
        }
return(FALSE);
}

/* This function compares names of cases. If the names are very close it
returns TIGHT. If the names match somewhat it returns LOOSE. otherwise
it returns FALSE. */

int     case_table_type::compare_names(full_case_info_type *x,
case_info_type *y)
{
char                    name[MAX_CASE_LEN];
streamoff               name_ptr;
case_name_ptr_type      name_link;
int                     match, temp_match;

match = FALSE;
name_ptr = y->names;
do {
        name_ptr_file.seekg(name_ptr, ios::beg);
        name_ptr_file >> name_link;
        names_file.seekg(name_link.case_name_ptr, ios::beg);
        names_file.get(name, MAX_CASE_LEN, '\n');
        if ((temp_match = namecmp(x->name, name)) != FALSE) {
              if (temp_match > match)
                    match = temp_match;
              }
        name_ptr = name_link.next_case_ptr;
} while (name_ptr != END);
return(match);
}
```

```
/* This function compares two strings. If all words in the shorter
string are found in the longer string in the same order, then it returns
TIGHT. If >= 3 words are the same in the same order it returns LOOSE. If
> %60 of words match it also returns LOOSE. Else it returns FALSE. */

int       case_table_type::namecmp(char *a, char *b)
{
char    n1[CASE_NAME_LEN], *n2, *temp;
char   *w1;
int       wd_mtch = 0;
int       wds = 0;
int       match = TIGHT;

if (strstri(a, b) != NULL)
          return(VERY_TIGHT);
if (strlen(a) < strlen(b)) {
          strcpy(n1, a);
          n2 = b;
          }
else {
          strcpy(n1, b);
          n2 = a;
          }
w1 = strtok(n1, ";: ,.-");
while(w1 != NULL) {
          if (words->real_word(w1)) {
                if ((temp = strstri(n2, w1)) != NULL) {
                      wd_mtch++;
                      n2 = temp;
                      }
                else {
                      match = LOOSE;
                      }
                wds++;
                }
          w1 = strtok(NULL, ";: ,.-");
          }
if (match && wds != 0) {
          if (match == TIGHT && wd_mtch > 1)
                return(TIGHT);
          if (wd_mtch >= 4 && ((float)wd_mtch / (float)wds > .8))
                return(QUITE_TIGHT);
          if (wd_mtch >= 1 || ((float)wd_mtch / (float)wds > .5))
                return(LOOSE);
          }
return(FALSE);
}
```

If the case matching filter finds no match (i.e., returns FALSE) for all candidate records, a new record is added to the database, as has been mentioned.

If, on the other hand, a match is found, any new information about the case from the current reference (e.g., a parallel citation to an unofficial reporter) is added to the record in the database 16 for the case reference. When information is added in this way to a case in the database, this newly extended case record is used as the search case for another search of the database, steps 224 and 226, using the same matching filter described above. If exactly one matching case is found, the current case record is merged with the matching case record, step 228; if more than one matching case is found, all matching cases are merged into one record in the database.

The database 16 so generated can be further processed to generate "ideal" case references that can be used in processing search queries, building hypertext links, and so on. Each case record in the database includes all variants of the case name that have been encountered and a use count for each variant. In one embodiment, the variant that appears most frequently is chosen as the ideal case name. In alternative embodiments, the ideal case name must also meet other criteria, such a having a particular form, such as "A v. B", or "In re C". The ideal citation form for a reference will be, in one embodiment, a standard form, such as a "blue book" or California Style Manual. In another embodiment, all known parallel citations are included in the ideal citation. This allows the preprocessor to provide alternate citations to a cited case even if such citations are not present in the case being viewed, if the alternate citation appears somewhere in the database of cases. In addition, in this process the form of case citations are corrected by the preprocessor so that they accord with generally accepted citations rules.

Statute citations are recognized by the preprocessor through template matching. The templates are largely based on conventional citation formats for the various statutes. Thus, unconventional methods of citing statutes by individual judges may affect the

accuracy of statutes extraction. A certain range of formats can be tolerated as different templates can be set up to deal with discrepancies in the format of citations to statutes. Citations in an unconventional format are converted to the appropriate generally-accepted format.

The use of matching templates relies to some degree on a knowledge of the jurisdiction of the case. For example, a reference to the "Evidence Code" in a California case means a different thing than identical reference in a New York case. These problems are taken care of by matching some templates only if the case belongs to certain jurisdictions. Also, matching can occur based on the use of a word such as "Act" or "Code" in the case (as in "Copyright Act" or "Bankruptcy Code").

References to sections (and subsections) of statutes are also identified by template matching. These references are associated with a statute primarily on the basis of the proximity of the section reference in the text to a reference to a statute.

Legal concepts are automatically extracted from text by matching sections of the text with terms contained in a legal concept dictionary, which is constructed by hand. The dictionary is a domain lexicon of words or phrases used by legal professionals. Each term in the dictionary consists of the stems of one or more words, which are matched to the text, and a legal concept, to which the stem is linked. Most concept phrases in the dictionary are associated with more than one stem, thus allowing users to retrieve documents containing terms that are synonymous or semantically similar to concept phrases selected as search terms by the user. The dictionary also distinguishes between entries that require an exact match versus entries that allow the matched information to appear in the text in any order, to be suffixed, or to be separated by noise words.

Concepts represent ideas. A legal concept generally gets its meaning from its relationship to a set of concepts which together constitute a legal doctrine or a procedural practice. Whether or not a word or phrase is, or represents, a legal concept ought to be measured in terms of its relationship to a particular doctrinal structure or sets of structures. For example "confession" is a legal concept because it has a semantic relationship to a doctrinal structure about responsibility and evidence, which in turn is part of a doctrinal sub-set of criminal law.

Since almost any kind of idea constitutes a concept of some kind or other, no sharp distinction can be drawn between the legal sub-language and the language of natural discourse. Some of the concepts of legal discourse are highly technical and unique to law, others have a great deal to do with the law, but are widely used in general discourse. Still other concepts are non-legal but are, in fact, extensively used in the discourse of law. The goal in constructing a concept dictionary is to include in it only concepts that function as signifiers in that they are generally correlated with factual concepts that function as the signified.

The legal concept dictionary is constructed from a wide variety of legal sources such as legal dictionaries, thesauri, statutes, indexes, learned authorities, and treatises. The dictionary includes synonym information and allows (when applicable) the matched information to appear in the text in any order, to be suffixed, and to have the words in the concept phrase separated by noise words.

As has been mentioned, a typical case contains a factual story, a description of the set of legal issues which the story gives rise to, a statement of the applicable law, and a resolution of the issues when the law has been applied to the facts. After the removal of legal concepts that are recognized by the legal concept dictionary, and of

case and statute citations that are recognized by template matching functions, the remaining text represents the facts of the case. Single word fact terms can be generated by removing noise words as determined by a noise word list. However, an improved indexing and query representation can be achieved by incorporating multi-word fact terms in both document and query representations. Unlike the recognition of legal concept phrases, which is dictionary based, the recognition of fact phrases is based on automatic sentence construct analysis. The underlying technology is described in the following references, whose disclosures are included here by this reference: Dillon, M. and Gray, S., FASIT: A Fully Automatic Syntactically Based Indexing System, *Journal of the American Society of Information Science* 34 (1983); Fagan, J., Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods, *Ph.D. Thesis, Technical Report* 87-868, Cornell University, Computer Science Department (1987); Smeaton, A., Information Retrieval Research: How It Might Affect the Practicing Lawyer, in S. Nagel, ed., *Law, Decision-making and Micro Computers* (1991); and Croft, W.B., Turtles, H.R., and Lewis, D.D., The use of Phrases and Structured Queries in Information Retrieval, in *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval* (1991).

The preprocessor recognizes multi-word fact phrases by combining term distribution and proximity information with a lexicon of noise words. It defines categories of noise words and uses them as "glue" connecting fact terms into phrases. "Joiners" (e.g., "by", "of") join two fact terms; "modifiers" (e.g., "extended", "civil") qualify or constrain terms; and "pure noise" is eliminated. The preprocessor also uses a set of rules to identify classes of noise terms such as names and numbers that are retained in the context of citations. While this simple approach to phrase recognition can result in some meaningless, useless terms, or over-specific terms, their occurrence is

minimized by the use of corpus filtering; i.e., the elimination of terms whose collection frequency falls below a given threshold and by applying constraints on the length of automatically determined terms.

Returning to FIGURE 1, with the concepts, case references, statutes, and facts extracted from a database of cases 15 -- e.g., one representing a body of case law -- the database of cases 15 can be searched with a query having terms of one or more types or categories (i.e., from one or more of the profile quadrants). The search and retrieval model that will be described is an adaptation of the vector space model, described in Salton, G., and McGill, M.J., *Introduction to Modern Information Retrieval* (1983), the disclosure of which is incorporated by this reference.

Turning to FIGURE 3, the retrieval model represents both documents and queries, step 310, as lists of weighted words and phrases. Relevant documents are retrieved by comparing a query composed of keyword terms of one or more types to document profiles stored in a database. At step 312, the candidate list of documents against which the query is tested is made up of those documents in which any of the search terms are found. (If the NOT modifier is used, the terms it modifies are not included in making the initial selection.)

At step 314, the similarity between a document and query is calculated with a metric based on the cosine formula, which is described in Salton, G., *Automatic Text Processing*, Addison-Wesley, 1989. The cosine formula measures the extent of a match between the document and query and produces a similarity score with respect to a document j, taking a normalized inner product between a query vector and a document vector, as follows,

where

$$sim_j = \frac{\sum_{i=1}^{n} (q_i \times d_{ij})}{\sqrt{\sum_{i=1}^{n} (q_i)^2} \times \sqrt{\sum_{i=1}^{n} (d_{ij})^2}} \quad (1)$$

sim_j = the calculated similarity of a query to a particular document j

n = the number of terms in database

q_i = the user assigned weight for term i

d_{ij} = the weight of term i in document j (this is a vector in i only)

and where

$$d_{ij} = f_{ij} \times \log\left(\frac{N}{n_i}\right) \quad (2)$$

where

f_{ij} = the number of times term i appears in document j

N = the total number of documents in the collection

n_i = the number of documents contain term i

This formula can be expressed more succinctly using conventional vector notation, where

Q = the vector of values q_i

D_j = the vector of values d_{ij}

by which the similarity measure set forth above can be expressed (step 322) as

$$sim_j = \frac{Q \cdot D_j}{|Q| \times |D_j|} \quad (3)$$

The vector space method and cosine formula were developed to measure the similarity of one document to other documents in a collection. The values q_i, in that context, were proportional to the number of times the i-th term appears in the document that serves as a query. In the present method, the values q_i have a default value of 1, but are user selectable to other values. User interface categories of High, Medium, Low, and Not simplify the user's task in selecting these values and are translated into the numeric values q_i used in the similarity calculation. The Not category actually represents a negative number, resulting in documents that contain the term to have a lower score than they would otherwise. Thus, the retrieval model allows the user to identify the importance of a term.

The retrieval model differs from the basic cosine formula (1), above, in other respects as well. The document weight factor is normally calculated in the retrieval model using the square root of the frequency factor, as follows, rather than formula (2), above.

$$d_{ij} = \sqrt{f_{ij}} \cdot \log\left(\frac{N}{n_i}\right) \quad (4)$$

Also, at optional steps 320 and 324, the model provides for a quadrant-based weighting factor k_h. (The subscript h selects one of the four types. just as the subscript i selected one of the terms in the database.) Thus, with quadrant-based weighting the similarity is measured by the type of the query term (that is, in the four-quadrant profile set forth above, by whether the query term is one of concept, case, statute, or fact), and the similarity for each type is weighted by the factor k_h. This is expressed (steps 322 and 324) in the following formula,

$$sim_j = \sum_h \frac{k_h \times Q_h \cdot D_{jh}}{|Q_h| \times |D_{jh}|} \quad (5)$$

where the subscript h indicates that the vectors Q_h and D_{jh} are limited to the terms in the respective quadrants.

Alternatively, the normalization is done on the document as a whole, thus.

$$sim_j = \frac{\sum_h k_h \times Q_h \cdot D_{jh}}{|Q| \times |D_j|} \quad (6)$$

(This is an example of normalization step 326.) Also, the retrieval model provides for the weighting of the similarity score by a user-selectable factor, step 328. The alternatives are (a) the factor 1, (b) the number of quadrants in which a match was found, (c) the square root of the number of quadrants in which a match was found, and (d) the ratio of the number of terms in the query found in document j divided by the total number of terms in the query. Alternative (d) scales the similarity score to the ratio of the total number of query terms found to the total number of query terms. Use of this alternative (d) is the default configuration. This emphasizes documents that have many matching terms over documents that have one term matching many times. Alternative (d) may be used in combination with either alternative (b) or (c).

The model also allows the user to scale the similarity score on a quadrant (type) basis with a factor that is the number of terms of the type (i.e., in the quadrant) in

the query found in document *j* divided by the total number of terms of the type in the query.

As can be seen, the retrieval model allows the user to weigh document search terms to reduce the importance of common words in a search while maintaining the importance of multiple hits within a document. By using the length of the query and the length of the document (measured by the number of terms that occur in each) to normalize the score, queries or documents that are particularly verbose do not tend to swamp the results.

The present invention has been described in terms of specific embodiments. The invention, however, is not limited to these specific embodiments. Rather, the scope of the invention is defined by the following claims, and other embodiments are within the scope of the claims. For example, the metric for ranking need not be the cosine formula, or a formula derived from the cosine formula. Other metrics, including a probabilistic metric, such as, for example, a Bayesian belief network, can also be used.

CLAIMS

What is claimed is:

1. A method for matching a current case reference to a set of case references, comprising:

(a) maintaining a database of the case references that have been processed by the method;

(b) parsing the current reference for its citations and, for each citation (the current citation) in the current reference, parsing the current citation into its volume, reporter, and page;

(c) searching the database of the case references for a matching case by applying a set of tests to the cases in the database of the case references, the set of tests including:

(A) if a candidate case matches the current case in two volume-reporter-page citations from different reporters, the candidate case is a match.

2. The method of claim 1, further comprising:

(d) parsing the current reference for its party names and, for each non-noise word in each party name in the current reference, acquiring a sound-alike value for the non-noise word;

(e) searching the database of the case references for a matching case by applying a set of tests to the cases in the database of the case references, the set of tests including:

(A) if a candidate case matches the current case in one citation, and the cases' names match at least loosely, and neither case has a

citation that is inconsistent with the citations of the other case, the candidate case is a match.

3. The method of claim 2, further comprising:

(f) initializing a candidate set to be empty;

(g) adding to the candidate set a case reference from the database if it has a same party name as the current reference, by sound-alike values;

(h) adding to the candidate set a case reference from the database if the case reference has a volume-reporter-page citation matching the current citation;

(i) searching the candidate set rather than the entire database of case references for a matching case by applying a set of tests to the cases in the candidate set.

4. A method for matching a current case reference to a set of case references, comprising:

(a) maintaining a database of the case references that have been processed by the method;

(b) initializing a candidate set to be empty;

(c) parsing the current reference for its citations and, for each citation (the current citation) in the current reference,

(1) parsing the current citation into its volume, reporter, page, court, and year, and then

(2) adding to the candidate list a case reference from the database if the case reference has a volume-reporter-page citation matching the current citation;

(d) searching the candidate list for a matching case by applying a set of tests to the cases in the candidate list, the set of tests including a test selected from the group consisting of:

(A) if a candidate case matches the current case in two volume-reporter-page citations from different reporters, and the court information for the two cases is not inconsistent, the candidate case is a match,

(B) if a candidate case matches the current case in two volume-reporter-page citations from different reporters, and the year information is not inconsistent, the candidate case is a match, and

(C) if a candidate case matches the current case in one citation, and both the court information and the year information for the two cases is not inconsistent, and neither case has a citation that is inconsistent with the citations of the other case, the candidate case is a match.

5. The method of claim 4, further comprising:

(e) parsing the current reference for its party names and, for each non-noise word in each party name in the current reference, acquiring a sound-alike value for the non-noise word;

(f) adding to the candidate set a case reference from the database if it has a same party name as the current reference, by sound-alike values;

(g) searching the candidate set for a matching case by applying a set of tests to the cases in the candidate set, the set of tests including a test selected from the group consisting of:

(A) if a candidate case matches the current case in one citation, and the cases' names match at least loosely, and neither case has a citation that is inconsistent with the citations of the other case, the candidate case is a match;

(B) if a candidate case matches the current case in one citation, and the courts and the years also match, the cases' names match very tightly, and the cases have less than two citations that are inconsistent from one case to the other, the candidate case is a match; and

(C) if a candidate case matches the current case in both courts and the years, the cases' names match very tightly, and neither case has a citation that is inconsistent with the citations of the other case, the candidate case is a match.

6. A method for ranking the relevance of a target document found by a search query in a set of documents, where the query has search terms, the method comprising:

(a) providing a set of weighting factors defined by a user for the search query, at least one of which weighting factors differing from the other weighting factors in the set of weighting factors and at least one weighting factor having a negative value, whereby each search term has an associated weighting factor;

(b) applying a metric function to the search query, the weighting factors, and the target document to produce a similarity measure for the target document against the search query weighted by the weighting factors; and

(c) ranking the target document by its similarity measure.

7. The method of claim 6 wherein the metric function includes a calculation of an inner product between a search query and the documents in the set over a document term vector space.

8. A method for ranking the relevance of a target document found by a search query in a set of documents, where the documents in the set of documents have document terms, each document term being of one of a plurality of types, and where the search query has search terms, each search term being of one of the plurality of types, the method comprising:

(a) for each type in the plurality of types, applying a metric function to those terms of the search query and the target document having the type, to produce a type-based similarity measure for the target document against the search query for each type in the plurality of types;

(b) combining the type-based similarity measures by applying a user-selectable type-based weight to each of the target document's type-based similarity measures to produce a final similarity measure; and

(c) ranking the target document by the final similarity measure.

9. A method for ranking the relevance of a target document found by a search query in a set of documents, where the documents in the set of documents have document terms, each document term being of one of a plurality of types, and where the search query has search terms, each search term being of one of the plurality of types, the method comprising:

(a) for each type in the plurality of types, applying a metric function to those terms of the search query and the target document having the type, to produce a type-based similarity measure for the target document against the search query for each type in the plurality of types;

(b) combining the type-based similarity measures and applying a factor to the similarity measures based on the number of different types for which some search term matched a target document to produce a final similarity measure; and

(c) ranking the target document by the final similarity measure.

10. The method of claim 9 further comprising:

providing a set of weighting factors for the search query, at least one of which weighting factors differs from the other weighting factors in the set of weighting factors, whereby each search term has an associated weighting factor; and wherein

the step of applying a metric function further comprises applying a metric function to the search query, the weighting factors, and the target document to produce a similarity measure for the target document against the search query weighted by the weighting factors.

11. The method of claim 10 wherein the metric function includes a calculation of an inner product between a search query and the documents in the set over a document term vector space.

12. The method of claim 10 wherein the step of providing a set of weighting factors comprises providing a set of weighting factors defined by a user for the search query.

13. The method of claim 12 wherein the step of providing a set of weighting factors comprises providing at least one weighting factor having a negative value.

FIG._1

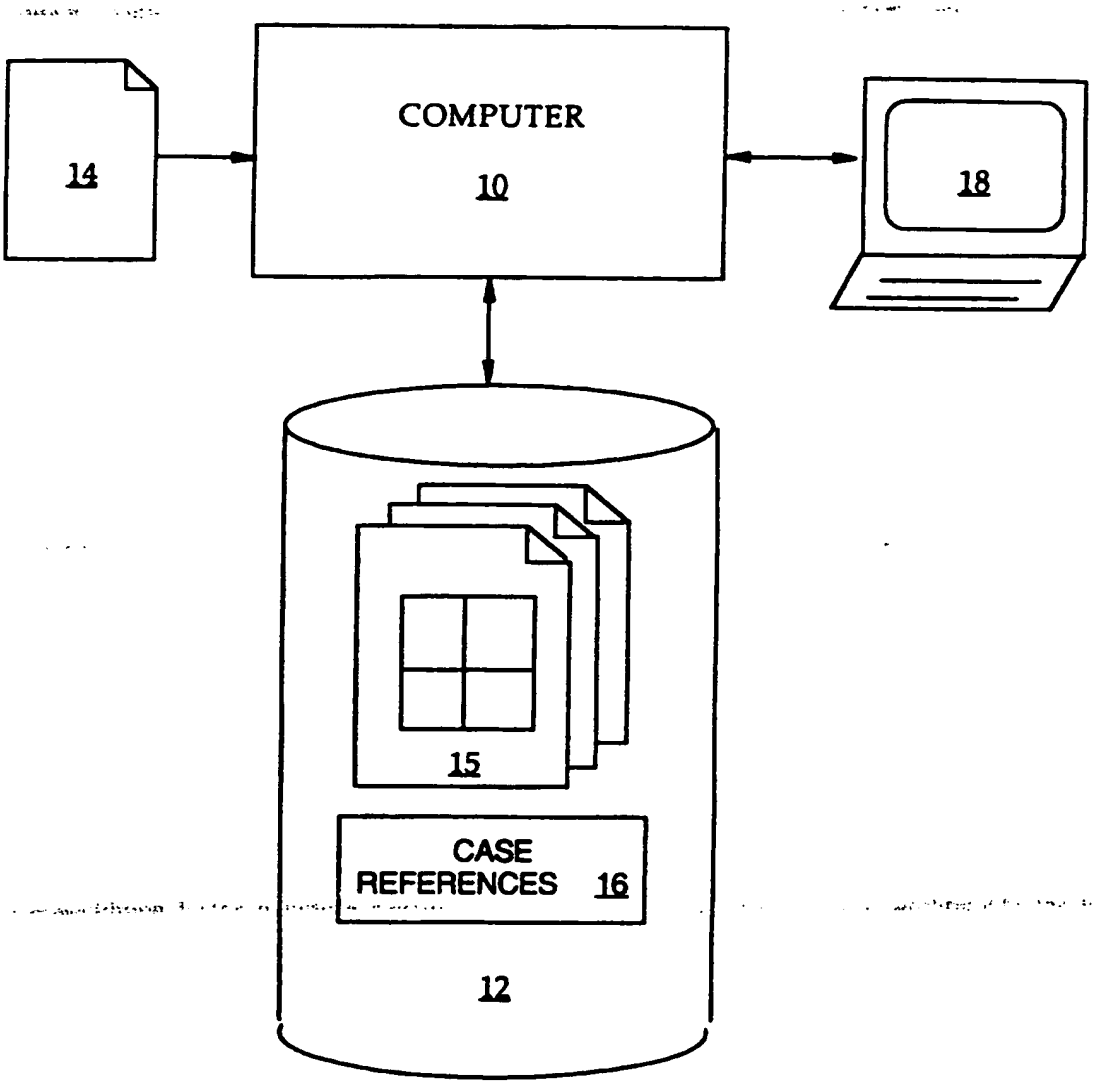


FIG._2

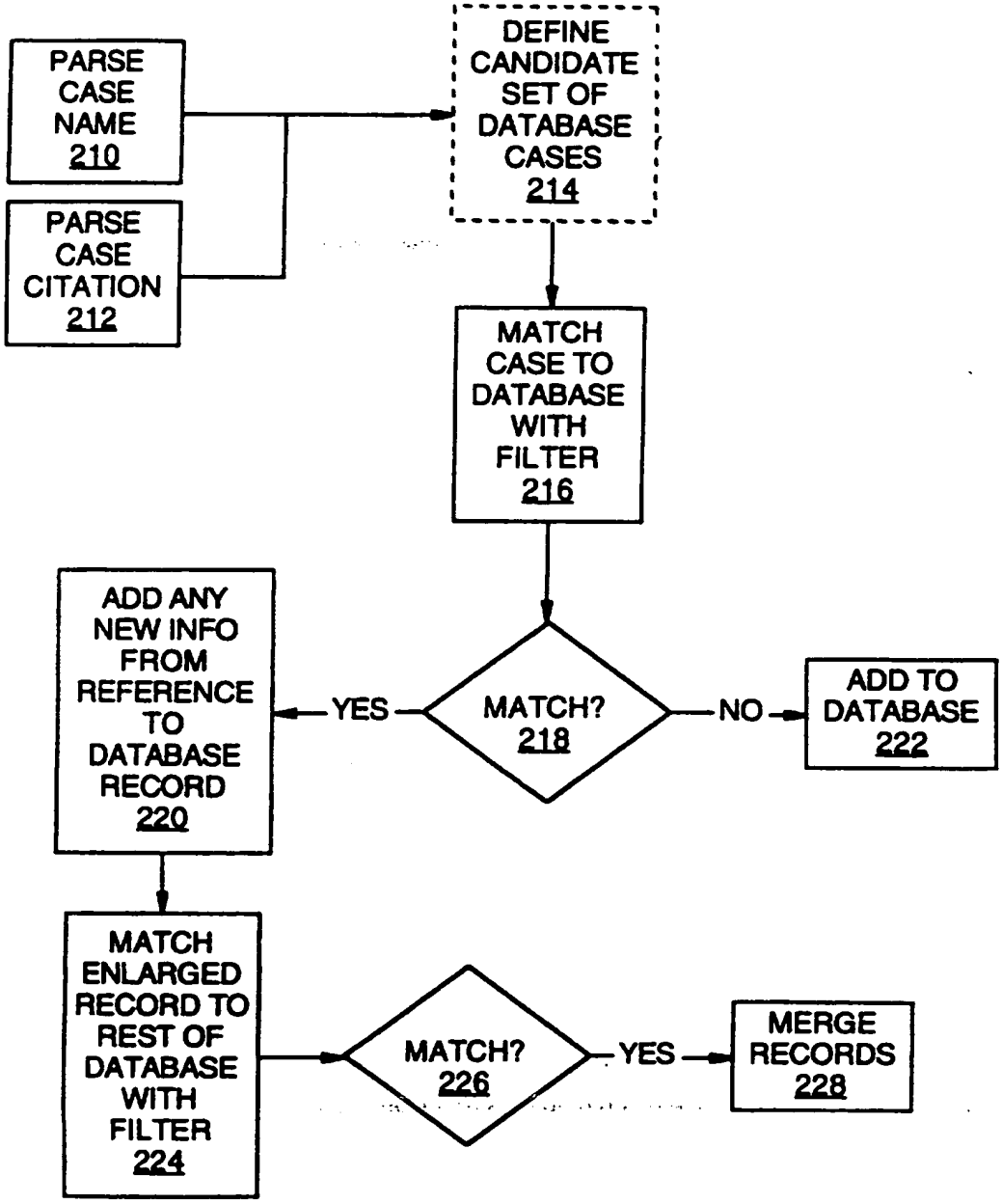
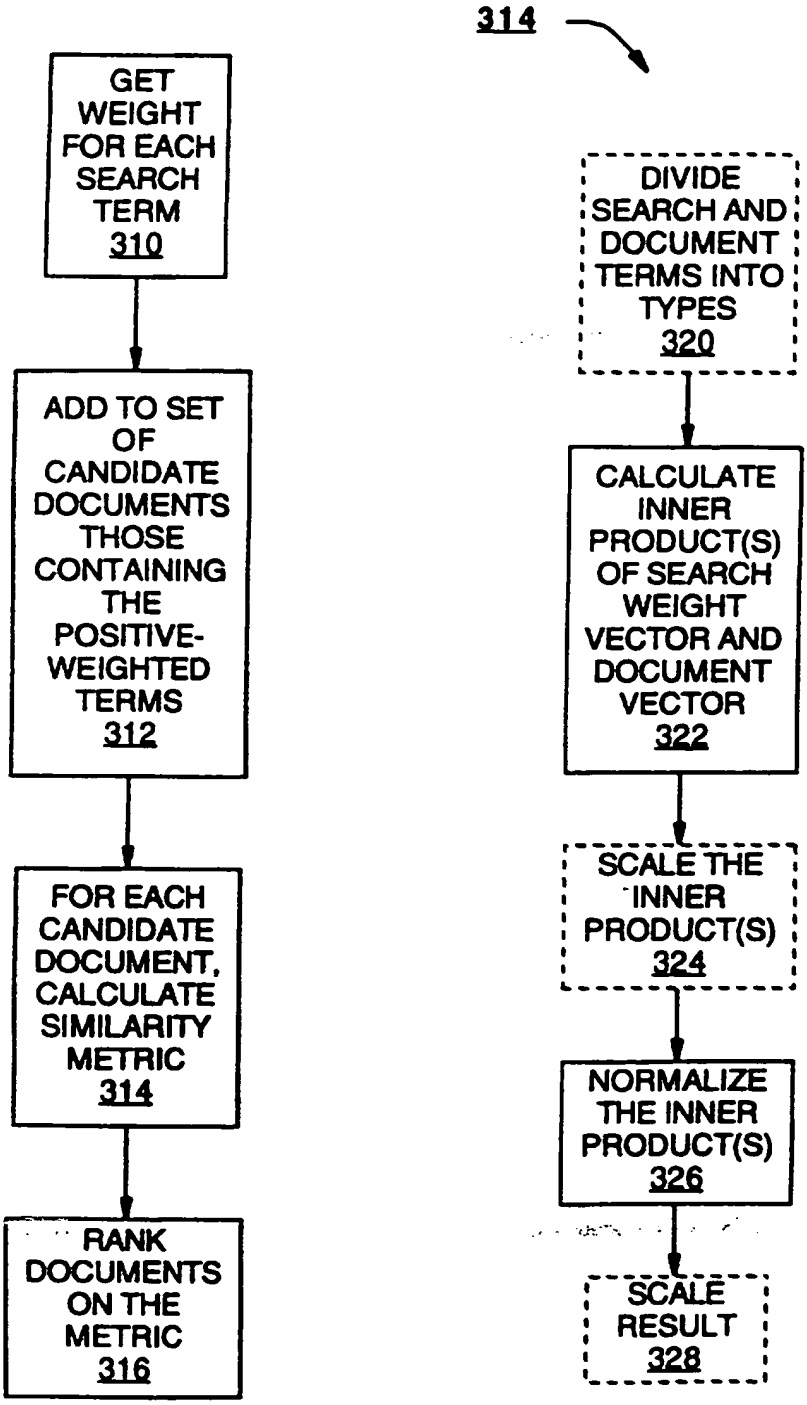

PARSE CASE NAME 210
PARSE CASE CITATION 212
DEFINE CANDIDATE SET OF DATABASE CASES 214
MATCH CASE TO DATABASE WITH FILTER 216
MATCH? 218
YES
NO
ADD ANY NEW INFO FROM REFERENCE TO DATABASE RECORD 220
ADD TO DATABASE 222
MATCH ENLARGED RECORD TO REST OF DATABASE WITH FILTER 224
MATCH? 226
YES
MERGE RECORDS 228

FIG._3


GET WEIGHT FOR EACH SEARCH TERM 310
ADD TO SET OF CANDIDATE DOCUMENTS THOSE CONTAINING THE POSITIVE-WEIGHTED TERMS 312
FOR EACH CANDIDATE DOCUMENT, CALCULATE SIMILARITY METRIC 314
RANK DOCUMENTS ON THE METRIC 316
314
DIVIDE SEARCH AND DOCUMENT TERMS INTO TYPES 320
CALCULATE INNER PRODUCT(S) OF SEARCH WEIGHT VECTOR AND DOCUMENT VECTOR 322
SCALE THE INNER PRODUCT(S) 324
NORMALIZE THE INNER PRODUCT(S) 326
SCALE RESULT 328

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/15624

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 17/30, 17/27; G11C 13/04
US CL :395/600; 364/419.19; 364.252.1
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/600; 364/419.19; 364.252.1

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, EIC, Law Library, DIALOG, DR-LINK

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,321,833 A (CHANG ET AL) 14 JUNE 1994 (29/08/90), see entire document.	6-13
Y	US 5,157,783 A (ANDERSON ET AL) 20 October 1992 (26/06/88), see entire document, especially Figs. 4, 7, 9-17, cols. 1, 27-32.	1-5
Y	US 5,278,980 A (PEDERSEN ET AL) 11 January 1994 (16/08/91), see cols. 1-12.	1-5
Y	COHEN et al. Finding the Law, An Abridged Edition of How to Find the Law, 9th ed. Minnesota: West Publishing. 1989	1-5
Y	JOHNSON-MALONEY et al. WESTLAW Reference Manual, 5th Edition. Minnesota: West Publishing. 1993, sections 4.7, 7, 10, 14, and 16.	1-5

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 31 OCTOBER 1996	Date of mailing of the international search report 02 DEC 1996
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer CHARLES RONES Telephone No. (703) 306-3030

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/15624

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,265,065 A (TURTLE) 23 November 1993 (08/10/91), see entire document.	6-13
Y	US 4,823,306 A (BARBIC ET AL) 18 April 1989 (14/08/87), see entire document, expecially Figs. 1a-b, cols. 1-8.	6-13
Y, P	US 5,537,586 A (AMRAM ET AL) 16 July 1996 (30/04/92), see entire document, especially Figs. 3-11, 14-15, col. 1-12.	6-13
X, P	US 5,544,352 A (EGGER) 06 August 1996 (14/06/93), see entire document, especially Figs. 4H-I, 5B-H, cols. 1-7 and 11-30.	1-13
A	ROSE et al. Legal Information Retrieval: A Hybrid Approach, International Conference on Artificial Intelligence and Law. Canada: ACM Press. 13 June 1989, see pages 138-146.	1-5
X	GELBART et al. Beyond Boolean Search: FLEXICON, A Legal Text-Based Intelligence System. England: ACM Press. 25 June 1991, see pages 225-234.	1-6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/15624

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Please See Extra Sheet.

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☒ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No: PCT/US96/15624

BOX II. OBSERVATIONS WHERE UNITY OF INVENTION WAS LACKING
This ISA found multiple inventions as follows:

I. Claims 1-5 , class (364/400-401), drawn to a specific method related to matching legal citations.

II. Claims 6-13, classes 395/(600,419.19), drawn to a method related to ranking and ordering various documents.

The species listed above do not relate to a single inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, the species lack the same or corresponding special technical features for the following reasons: Group I is related to matching of legal citations in databases, whereas Group II is related to ranking and ordering the similarity of documents.
The matching of legal citations as per group I employs a different technique (special technical feature) then that employed to rank and order the smiliarity of documents in that the group I claims include the parsing of a case reference for its citations and performing a search for an appropriate match, whereas group II provides weighting factors defined by the use of a search query and applies a metric function to the search query, the weighting factors and the target document.